ロボットのソフトウェア : tutorial(ソフトウェア科学・工学の数理的方法)

AUTHOR(S):

塚本, 享治; 末広, 尚士

数理解析研究所講究録
511巻 1984年 97-119

ロボットのソフトウエア (tutorial)

電子技術総合研究所　塚本享治 (Michiharu Tsukamoto)
電子技術総合研究所　末広尚士 (Takashi Suehiro)

1。 まえがき

ロボットは環境に働きかける腕や足などの効果器、環境情報を収集する目や耳などの感覚器、およびコントローラとしての計算機が一体化されたものであり、機械単独や計算機単独では持ち得ないさまざまな潜在能力を有している。潜在能力の第一は、ハードウエアとソフトウエアが分離されており、ロボット治具や対応するソフトウエアを交換することによって、さまざまな作業に対処できることである。第二は、センサを通して外界を認識することによって、環境に適した行動を行なうことができること。第三は、アクチュエータを使って環境に働きかけることによって、より良い状況のもとで認識や作業を行なうことができること。第四は、計算機同士の通信によって空間的な広がりを持つことができること。最後に、ソフトウエアを媒体として、過去の遺産を受け継ぐとともに、将来の発展性にも富むことが挙げられる。しかしながら、現実のロボットにおいてはこれらの潜在能力が十分に発揮されているとは言い難く、潜在能力の多くがソフトウエアに関するものであることから、各方面でロボットのソフトウエアに関する研究と開発が精力的に進められている。

ところで、ロボットのソフトウエアに関する研究はここにきて２つに分化する傾向にある。１つは、産業用ロボットによる工場の自動化を目指すものである。メーカや米国の研究機関において活発に研究と開発が行なわれており、計算機関係の雑誌の最近の特集号 [Computer82a,Computer82b,Proceedings83]によってもその概要を知ることができる。もう1 つは、人の近づけない環境で人に代わって作業を代行する遠隔作業ロボットに関するものであり、わが国のロボット研究者のここ数年の課題となっている[ICAR83]。

本稿では、後者の遠隔作業ロボットのソフトウエアについて概説を試みる。まず２節で産業用ロボットと遠隔作業ロボットとの関連、及びロボットの基礎知識について述べる。つづいて、３節でロボット要素をプログラミングするためのロボット言語、4節でロボットと人とのインタラクションについて述べ、最後に５節で、全体を統合し統括するシステムについて述べよう。

2。　ロボットの基礎知識

2。1　産業用ロボットと遠隔作業ロボット [Tsukamoto83]

現在、最も数多く使われているロボットは、工場の製造ラインなどに組み込まれる産業用ロボットである。工場は人工的な環境であり、しかも一度システムを組むとある程度の期間そのまま使用できる。そこで、製造ライン、部品フィーダ、治具などをロボットに合わせて整備するというアプローチがとられている。位置情報だけで作業が遂行できるように、所定の位置に所定の物が存在するように環境を整備するのである。しかしながら、位置情報だけでは、ハメアイ作業や不確実な環境での作業は困難である。そこで、ロボットの手先などにセンサを付け、わずかではあるが環境に適応させるという解決法がとられている。

一方、工場の外では、作業環境に未知な部分が多く、また仕事も単発的で多岐に渡っているので、産業用ロボットのアプローチは通用しない。このような環境で行なわれているのは、ロボットを逐一遠隔から操縦(guide) するというアプローチである。すべての操作と判断はオペレータが行なうことになるので、オペレータの技能に依存するところが大きく、またオペレータの精神的、肉体的な負担は極めて大きなものとなる。これらを解決するために、直接的な行動や感覚によってロボットを制御するのではなく、行動や感覚を抽象的な記号に置き換え、あらかじめシステムに記憶してある作業環境や作業手順などの知識をもとに、わずかな指令でロボットを知的に遠隔操作することが試みられようとしている。

2。2　環境モデル

ロボットを知的にするには、何がどこにあるのかという大局的な情報が不可欠である。これは、ロボットがその内部にロボットの置かれた環境のモデルを持つべきことを意味している。現在のロボットでも大体の位置と姿勢がわかれば、ローカルなセンサなどによって手さぐりで作業することが可能となっており、環境のモデルが実環境と寸分たがわぬものである必要はない。環境モデルはロボットが思い描いている現実の世界を抽象化したものである。

現実の環境に存在する実体は、環境モデルの存在物に対応し、プログラムの世界ではデータオブジェクトに対応する。プログラムによる扱いを容易にするために、環境モデルの中の各対象物に、①名前、②位置と姿勢、③関連ある対象物との関係、④その他の情報、を持たせ、名前で区別でき指定できるようにするのは極めて自然な発想であろう。また、各対象物はいくつかのカテゴリに分類でき、それに対応した一般的な性質を持ち、取扱われるべきである。これらは、環境のモデリングには抽象データ型あるいは対象指向型のアプローチが適していることを示唆している。

環境モデリングで重要なのは、位置と姿勢、それに関係ある対象物との関連であろう。位置とはその対象物が規準座標系からみてどこにあるかであり、姿勢とはその対象物がどの方向を向いているかである。環境の規準座標系を Σ env とし、対象物にそれぞれの座標系 Σ obj を設定する。対象物の位置は Σ obj の原点が Σ env のどこにあるかで指定でき、姿勢は Σ obj の座標軸が Σ env とどのような関係にあるかで指定できる。この座標系の間の関係をフレームと呼ぶ。ここで、フレームは実体のある対象物に限る必要はなく、把握点や目でみる際の着目点など架空のものにも設定して良い（図2.1)。ロボット自身も環境モデルの中の一つの対象物である。フレームに関し、もうひとつ重要なのは、例えば、部品をネジどめすれば、一方を動かせば他方も自動的に動くといった相互の関係である。

このようなフレームの概念の導入によって、環境が抽象的かつ定量的にモデリングできる。その結果、例えば、マニピュレータによる対象物の把握作業は、『マニピュレータを、手先きのフレーム Σ handが対象物のフレーム Σ obj に一致するように動かし、指を閉じる。』といった抽象的な指示やプログラミングが可能となる。次に、このことをもう少し定式的に述べる。

2。3　ロボットの幾何学

空間内の剛体の運動は並進運動と回転運動から成り、それらは線形な座標変換として扱うことができる。スケール変換を伴わない3次元空間内の回転変換と並進変換はまとめて、運動座標系を基準座標系に変換する次のような4×4の変換マトリックスTで表現できる。また、3次元空間内の点の位置を表わすベクトルvはこの記法では $\begin{bmatrix} v \\ 1 \end{bmatrix}$ と表わされる。尚、右手座標系、縦ベクトルである。

$$T = \begin{bmatrix} R_{3x3} & p_{3x1} \\ 0\ 0\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \text{Rotation} & \text{Position} \\ 0\ 0\ 0 & 1 \end{bmatrix} = \begin{bmatrix} xi & yi & zi & p \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

この記法を用いると、x軸回りの回転α、y軸回りのϕ、z軸回りの回転θ、原点の並進移動 $(dx,dy,dz)^t$ に対応する変換マトリックスは次のようになる。

$$T_{x,\alpha} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C\alpha & -S\alpha & 0 \\ 0 & S\alpha & C\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{y,\phi} = \begin{bmatrix} C\phi & 0 & S\phi & 0 \\ 0 & 1 & 0 & 0 \\ -S\phi & 0 & C\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_{z,\theta} = \begin{bmatrix} C\theta & -S\theta & 0 & 0 \\ S\theta & C\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_{trans} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ここで、$S\alpha = \sin\alpha$,　$C\alpha = \cos\alpha$

変換される座標系（座標系ｊから基準座標系ｉ への変換）を明示する必要のあるとき、T_i^j と記述する。この変換マトリックスTが先に述べたフレームである。変換マトリックスの積はまた変換マトリックスであるから、このことは変換マトリックスによってフレームの基準座標系が変換されても良いことを意味している。

（１）環境

図2.1 おいてテーブルを基準座標系としたＴＶカメラのフレームを T_{table}^{camera} とする。ＴＶカメラを通して見たテーブルの基準座標系とテーブル上の箱のフレームをそれぞれ、T_{camera}^{table} , T_{camera}^{box} とすると、テーブルの基準座標系に対する箱の位置と姿勢は

$$T_{table}^{box} = T_{table}^{camera} \cdot T_{camera}^{box} = (T_{camera}^{table})^{-1} \cdot T_{camera}^{box}$$

このように、フレームを用いると環境モデルの取り扱いが容易である。

（２）マニピュレータ [Lee82]

マニピュレータの関節ｉの座標系ｘi-1, ｙi-1, ｚi-1 を次のように定める。①関節ｉの動きの軸をｚi-1 軸、②ｘi-1, ｙi-1, ｚi-1 座標系の原点はｚi-1 軸とｚi 軸の交点、またはｚi-1 軸とｚi 軸の共通法線とｚi-1 軸の交点、③ｚi-1 ×ｚi 方向、またはｚi-1 軸とｚi 軸の共通法線方向にｘi-1 軸，④右手座標系となるようｙi-1 軸。さらに、ｚi-1 軸回りのｘi-1 軸からｘi 軸への角度を θ_i 、関節i-1 と関節i の距離のｚi-1 軸成分を d_i 、ｘi 軸成分を a_i 、ｘi 軸回りのｚi-1 軸からｚi 軸への角度を α_i とすると、

$$A_{i-1}^{i} = T_{z,\theta_i} \cdot T_{z,d_i} \cdot T_{x,a_i} \cdot T_{x,\alpha_i}$$

このとき、関節によってリンクがシリアルに結合されたマニピュレータの変換マトリックスは次のようになる。

$$T_0^i = A_0^1 \cdot A_1^2 \cdot \ldots \cdot A_{i-1}^i = \begin{bmatrix} x_i & y_i & z_i & p_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

実際のマニピュレータ、例えばＰＵＭＡでは、座標系は図2.2 のようにとることができ、定数である α_i に所定の値を代入すると、各変換マトリックスは図2.3aに示すようになり、これでマニピュレータの機構が定まる。実際にマニピュレータを動かすには、各関節の角度 θ_i を決めなければならない。それには、手先の位置と姿勢Ｔが与えられて、次の式を θ_i について解くことになる。

$$T_0^6 = A_0^1 \cdot A_1^2 \cdot \ldots \cdot A_5^6$$

これは一般には解けないので、ヒューリスティックスが必要である。

まず、リンク３の先端の位置に着目すると、第一の等式が成り立ち、さらにこれはT_0^4 の第3 列の一部であるから、第二の等式が成り立つ。

$$\begin{bmatrix} px \\ py \\ pz \end{bmatrix} = p6 - d6\ a = \begin{bmatrix} C1(a2C2+d4S23)-d2S1 \\ S1(a2C2+d4S23)+d2C1 \\ d4C23-a2S2 \end{bmatrix}$$

これを解いて$\theta 1 \sim \theta 3$ が求まる。次に手先の姿勢に着目して、

$$z4 = \pm(z3 \times a)/|z3 \times a|, \quad a = z5, \quad s = y6$$

を解くと$\theta 4 \sim \theta 6$ が求まる。$\theta 1 \sim \theta 6$ は図2.3bに示す通りである。ここで、$\theta 1 \sim \theta 3$ の±は、それぞれ、肩の左右、手に対する肘の上下、第３リンクの左右回転に対応している。

以上によって、対象物の位置と姿勢が与えられれば、そこに手先を持っていくことができる。しかし、現実のマニピュレータでは、途中で腕が物にぶつかったり、動きがぎくしゃくしないように、様々な配慮が払われている。

３。　ロボット言語

ひとくちにロボット言語と言っても、それを使う立場によって考え方も大きく異なる。ロボット言語の使用者は大きく２つに分けられる。１つはロボットのユーザでありもう１つはその開発者である。ユーザはロボットに仕事をさせることに関心があるので、ユーザ向けの言語ではどれだけ簡単な命令で役に立つ仕事をさせることができるかに重点が置かれる。一方、開発者はどのようにしてロボットにその仕事を実行させるかを問題としているので、開発者向けの言語ではロボットの細かい制御方法まで記述できる必要がある。

現在までのロボット、特に産業用ロボットは、塗装、溶接、運搬などあまり高度な技能を要求されない作業に適用されている。すなわちロボットは基本的には移動するだけでよく、どのようにして動作を実現させるかということは問題にされなかった。したがってロボット言語もいかに簡単にその軌道を表現するかということに重点が置かれ、複雑な動作を記述する言語は必要とされなかった。つまり、これまでのロボット言語のほとんどがユーザ向けのものと言うことができる。

しかし、ロボットの適用分野を簡単な産業用ロボットの分野から広げようとするにつれてより複雑で高度な技能が要求されるようになってきている。またロボットのハードウェアとしても、手先の位置だけでなくその力やコンプライアンスまで制御することのできるトルク制御マニピュレータや、外界の情報を取入れ利用するための各種のセンサが開発され、高度な技能を実現する手段も生れてきている。ここにおいてロボットの技能を実現するにはどのように制御したらよいかが問題とされ、それを記述することのできる言語の必要性も高まってきている。なお、この節に現われるロボッ

ト言語で参考文献に挙げていないものについては、[Bonner82]や[Lozano-Perez83]から孫引きされたい。

3。1　ユーザ向けロボット言語

ユーザから見たロボット言語は記述の容易さの観点から、①動作レベル、②対象レベル、③作業レベル、の3種に分類できる。

（1）動作レベル言語

ロボットの動作を逐一記述するレベルの言語で、現在実際にロボット言語として用いられているもののほとんどがこのレベルの言語である。腕の関節角を直接指示するタイプから、その手で持った物の動きを指示すればよいタイプまでいろいろある。WAVE, SIGLA, VAL, EMILY, PAL, AL . などがこの範中に属する。

（2）対象レベル言語

作業対象の状態や動作の記述によってロボットを動かそうとするレベルの言語で、その記述のレベルは、作業手順書程度である。このレベルの言語でも「ネジをはずす。」といったレベルから「ネジ回しを取り、その先をネジンにあて、ネジ回しを回し、ネジを取る。」のようなレベルが考えられる。衝突回避、軌道計画などを計算機で行うことも考えられている。このような言語としてはRAPT、AUTOPASS、などが提案されているが、まだ実用にはなっていない。

（3）作業レベル言語

「ある物を組立てる。」などの仕事が命令の単位となるもので、プログラミング言語というよりコマンド言語に近いと考えられる。このレベルの言語では、あらかじめ決められたことだけを実行するのではなく、ロボットの置かれた環境を理解し、それに応じて必要な動作の計画を行い、また途中で予期しない障害が生じたときにはそれに対処する必要があるとされている。実際のシステムとしてはSHAKEYなどの人工知能の分野で開発されかものがこれに相当するが、今後の研究に待つところが大である。

AL　[Mujtaba79]

スタンフォード大で開発されたものでそのシンタックスはAlgol 風である。図3.1 はALのプログラム例である。特徴は、座標系と座標変換に対応するデータタイプとしてFRAME とTRANS を導入し、前節で述べたように、腕の動きを関節角でなく、手先の座標系や物の座標系がある座標系から他の座標系へ動くという形式で腕の動きが記述できることである。その利点は次のようなものである。

①動作の記述が腕の機構と形式上独立になった。

②座標系を用いて記述する環境モデルと整合性が良く、その導入に成功した。

③動作に対して幾何的演算が可能なため、繰返しやサブルーチンが書き易い。

この記述方法は、それ以後のロボット言語に大きな影響を与えている。さらに、力の制御や動作条件のモニタリング、2 本の腕の協調動作などの機能も組込まれている。

ＶＡＬ [Simano79]

実際の産業用ロボットであるPUMAに用いられていることで有名である。シンタックスはアセンブラ風であり(図3.2)、環境モデルを持たないが、腕の動きはALのように座標系で指示でき相対座標による表現も可能になっている。さらに視覚からのデータの取込みも可能である。 VALの成功により最近では各ロボットメーカが自社ロボット用のロボット言語の開発を行っており、またそれらの標準化も話題に昇るようになってきている。

３．２ 開発者向けロボット言語

高度な技能を発揮し複雑な仕事を遂行するロボットのシステムは腕や足などの動作部、視覚や触覚などの感覚器、プランニングやデータベースまたは人間などの知能、知識などの各要素を有機的に統合したものである。したがってそのようなロボットを有効に働かすためには、それぞれの要素をプログラムできること、またそれらの統合された動作をプログラムできることが必要となる。システムの有機的な統合を実現するためには、全体がそれをサポートするOSの上にのった一つの言語としてまとめられでいることが望ましい。各要素の機能はその要素言語でプログラミングされることになる。ここではそのようなシステムのマニピュレータ言語と視覚言語を中心に説明する。このような言語の例はAMLである。

ＡＭＬ [Tayler82]

IBM で開発されたロボット言語で、その特徴は、インタプリタベースで動く多少ロボット向きの汎用言語にロボット用の関数が組込まれていることである。図3.3はプログラム例である。重要な点は、

①ロボットのプログラムの内で純粋にロボットの動作に関する記述の占める割合はそれほど多くないという観察に基づいて、汎用言語としての機能を十分にもたせている。

②動作のための関数は関節角目標値を出力するだけのもので、それを用いて細かい動作まで書ける。

③スーパセットとして視覚処理言語（AML/V ）を持つ(図3.4)。

④関数ベースなので新しい関数を作るだけで、新しいデバイスにも容易に対応できる。

などである。しかし，①知能や知識との結合が考慮されていない、②動作命令が関

節角を用いているため視覚やデータベースとの整合性がよくない、などの問題がある。特に、②はIBM のマニピュレータがXYZ 型であるためであり、腕の機構に依存しない汎用的な言語ではない。

電総研では現在LISPの方言であるPETLを用いて開発を行っている。これはロボットのさまざまな機能を簡単にインテグレートできるし、プログラム開発の効率もよいからである。

マニピュレータ言語

作業に応じた座標系を導入し、その座標系内での位置や姿勢、力やコンプライアンスを制御することはロボットの高度な技能を実現するためには不可欠である。そこで、それに適したトルク制御マニピュレータ[Takase83]を標準的なマニピュレータとして想定し、その機能が十分に記述できるようなマニピュレータ用の要素言語を開発中である［末広83］。図3.5 は、この言語をもとに作ったユーザ向けの言語によるプログラム例である。環境や作業のデータベースとの整合性などの配慮もなされている。

視覚言語

RVL は三次元物体を扱う視覚言語である[Matsushita83]。図3.6 、図3.7 はプログラム例である。その特徴を以下に示す。

①投射スリット光のステレオ視によって得た物体の距離断面（距離プロフィール）を利用する。

②距離プロフィールでの物体の見え方に基づいて、物体をモデル化する。

③モデルを利用して物体の照合、解析を行う。

このようにRVL では得られたデータを物体のモデルと結びつけることにより、データ入力装置および物体認識装置の両方の機能を言語によって記述できるようになっており、より高度な機能が実現できそうである。

4。ロボットへの教示

ロボットの世界では、人には簡単に思えるがプログラミングすると煩雑になることが多い。プログラムの結果は人には極めて直感的なロボットの動きであるのに対し、そのもととなるプログラムやデータが異質な論理や3次元の数値として与えられねばならないからである。こうしたものに、動作、3次元環境、仕事などがある。これらは、teach-by-doingやteach-by-showingによってロボットに直接教えるのが直感的である。

4。1　動作教示

最も簡単な動作教示はプレイバック方式である。プレイバック方式とは教示者がマ

ニピュレータの手先をつかんで所定の動作を行い、そのときのマニピュレータの関節角などの値を記録しておき、その記録に基づいて同じ動作を再現して作業させるというものである。この方式は直感的でわかりやすく、また比較的簡単に実現できるので早くから実用化され現在も多くの産業用ロボットに用いられている。しかしこの方式には、次のような欠点がある。

①動作起点や動作の一部の変更の際にも、すべての動作の再教示が必要である。

②動作の開始時と終了時以外には他の装置と連係動作させることができない。

③力、コンプライアンスなどを制御できる高度な腕が有効に生かせられない。

これらの問題点を解決するためには言語によるプログラミングが必要となる。

ロボット言語による動作の記述はそれ自身動作の教示と言うことができる。言語を使えば必要な動作は理屈の上ではすべて机の上で教示できる。しかし、作業に必要な位置や力などのすべてのデータを関節角、座標などの数値で直接書くことは難しい。そこで、実際に腕を動作させて必要なデータを収集しておき、実行時にはその値を利用することになる。多くの産業用ロボットではこのような機能が言語とセットにされている。

この場合に位置のデータを関節角ではなく座標系などの一般性のある記述を用いると、位置を変化させてのくり返し動作などのときにすべてのポイントを教示する必要がなくなる。さらにこれを環境モデルと結合すると一つの部品の一点を教示するだけでその部品の他の点は教示しなくてよくなるし、またその教示には腕を直接動作させずにレーザポインタやＴＶなど他の装置を用いることが可能になる。ここにおいて動作の教示は環境の教示へとつながっていく。

４。２　環境教示

動作教示に附随して、それに必要な環境情報が教示できるが、これらは環境情報のごく一部であって、軌道、視覚、行動計画などより高度な処理のためには不十分である。対象物の形状、位置、姿勢などをより正確に教示する必要がある。

産業用ロボットの扱う対象物は、部品やそれらが組み立てられた半完成品であり、それらに対しては、通常、部品や半完成品の図面がその製品の設計段階で用意されている。これを何らかの方法でシステムに入力すれば、対象物の形状を教示したことになる。最近のＦＭＳでは、これをさらに進め、計算機を使ってＣＡＤによって設計した製品の組立や検査をロボットに行なわせるためのロボットのプログラムをＣＡＤの情報をもとに自動作成しようとしている。

極限作業ロボットでは、図面を使うアプローチは一部には使えても、これだけというわけにはいかない。環境にバリエーションが多過ぎるので、現場で環境を教えるこ

とも必要である。この目的に沿うものに、レーザポインタによる位置計測機能とスーパインポーズディスプレイによる実環境とそのモデル線画との重ね合わせ表示機能との組み合わせによる環境教示方法がある[Hasegawa82]。図4.1 はそれを実現するシステムの構成である。

オペレータは実環境を映しているモニタテレビを見ながら、レーザポインタを操作して対象物の特徴点にレーザビームを照射する。物体上に生じたスポットをＴＶカメラで受け、三角測量によって、その点の３次元座標が計測される。必要な数だけ特徴点を取り込むと、システムに用意された基本立体を対象物にあてはめ、その位置、姿勢、形状、大きさを得る。システムが得た環境モデルの線画は、実環境を映すモニタテレビでスーパインポーズされて、表示される。オペレータは、この重なりの様子を見てモデルの良し悪しを知り、修正を繰り返してより正確な環境モデルにしていくのである。

対象物は、基本立体の組み合わせで表現される部分立体に対応するデータセルの組み合わせで表わされる。データセルの内容は、

VOL.　体積　　実物体は+,穴は- 、座標軸は0

FOR.　部分立体名

DIM.　部分立体の寸法

FRA.　部分立体に固定された座標系

AFXT. 含まれる親物体の名称と接続関係

AFXB. 含む子物体の名称と接続関係

図4.2 は記述例である。実物体で、名称はＢＲ、縦横高さはa,b,ｃ、重心の位置は（Px,Py,Pz）、固定された座標系の軸のベクトルは(Xx,Xy,Xz),..,(Zx,Zy,Zz)である。LBODY1とLBODY3が和、LBODY2が差の形で関係ずけられており、T はその結合が固定であることを意味し、Ｔで関係づけられた立体の一部を動かすと全体が動く。

５。　統合アーキテクチュア

ロボットのハードウエア要素とソフトウエア要素が充実されつつある現在、それらを組み合わせてより高度なロボットシステムを構築したいという要求が高まっている。特に、FMS を指向する分野で強いが[Tayler83]、これは遠隔作業ロボットでも同じである。ここでは、これまでに述べてきたことのいくつかを背景にし、電総研で現在開発を進めている統合システム[Tsukamoto83] について述べる。

ハードウエア的には、ロボットを制御するコントローラがネットワークで結合された分散システムである。システムの統合と統括の基本的な考え方は『未知な環境でも

、可能な範囲で自動的あるいは自律的に動作し、わからない部分や複雑な局面になると、判断を上位の物に委ね、最終的にはオペレータの教示や操縦に任せる』という考え方である。環境が未知ならほとんど操縦することになるが、環境が分ってきたり、プログラムが局所的であっても明確な範囲で知的になれば、段々にオペレータの手を離れて、自律的、並行的、かつ分散的に作業できるようになるであろう。これは、ロボットシステムの開発から運用までを連続な過程としてとらえるものである。

5。1　ロボット用ワークステーション

先に述べた考え方は複数のロボットとオペレータが一体となってインタラクティブに、ロボットシステムを開発し、また作業を進めるものである。オペレータが対峙するネットワークの1つのノードをロボット用ワークステーションと呼ぶ。ロボット用ワークステーションは、ロボットシステムの開発、操作、運用に必要な道具を一式取り揃えており、ロボットシステムの開発段階ではワークベンチ、開発されたロボットシステムを使って作業をする際には、操作台や監視台の役目を果たす。図5.1 のようなマルチメディアディスプレイとよぶ特殊なディスプレイと操作と教示のためのスティックを備えたノードである。

（1）マルチメディアディスプレイ

ロボットシステムの開発中や操作中には、作業環境全体、左右から見た作業状況、複数台のロボットの作業状況、プログラムからのメッセージなど、同時に複数の情報を簡潔に見たいことが多い。このような場合に、一画面上に簡潔に表示するのがマルチメディアディスプレイである。ＴＶカメラからの実際の環境や作業状況だけでなく、グラフィクスによってシステム中の仮想環境や仮想作業状況も表示できる。表示方式としては、ウインドウを単位として、画面の拡大・縮小、スーパインポーズ、立体視などが可能である。

（2）操作教示用ステック

マルチメディアディスプレイが簡潔な表示装置であるのに対し、操作教示用ステックはいわゆるマスタアームで、システムの簡潔な操作と教示のための装置である。システムに3次元の位置や姿勢、軌道、対象物などをオペレータが行為を通して直接に教えるものである。また、マルチメディアディスプレイを見ながら、実環境だけでなく、仮想環境でも、動作や作業の教示、対象物の指示などがその世界に入った雰囲気で行なえる。

5。2　ロボットオペレイティングシステム

ロボットシステムの開発からそれを使った作業までをサポートするロボットオペレイティングシステム(ROS) は、ロボット言語と同じくエンドユーザ向けと開発者向け

の最低２種類の顔を持つ必要がある。開発者向けには、ロボットシステムの開発に必要な道具を取り揃え、様々なシステムコンフィギュレーションに容易に対処でき、しかも小回りのきく開発環境を提供し、一方、エンドユーザ向けにはロボットシステム開発者の開発したロボットハードウエアとソフトウエアがふんだんに用意された知的で使い易い操作環境を提供する。ユーザと開発者の関係は相対的なものであるから、いろいろな意味で、柔軟なインタフェイスを持った統合性の良いシステムを作ることに尽きる。統合性には、①計算機システム、②ロボットシステム、③ユーザをも含んだシステム、の３レベルが考えられる。なお、図5.2 はオペレイティングシステムの機能構成図である。

システム全体を構築する上でその底流に対象指向型アプローチを置いている。①モデルの世界の記述と操作が自然である、②ロボットの雑多な知識の表現と活用に適する、③メッセージ通信が分散プログラミングに適する、④情報隠蔽やモジュール性などが大規模プログラミングに適する、⑤ハードウエアやシステムなどの広範な範囲がカバーでき、プログラムの分散化、ハードウエア化などが自然に行なえる、などがその理由である。

（１）計算機システムレベルの統合性

ネットワークで結合された分散システムであるから、ハードウエア上の統合性は良い。ソフトウエアの統合性のために、①ローカルカーネル、②分散カーネル、③分散システム記述言語、の３層が設定されている。

ローカルカーネルは、通常のカーネルと大差はないが、機種の違うノードに対し移植性と移動性を高めるために、仮想計算機を提供する。分散カーネルは、各ノードのローカルカーネル上に搭載され、ノード間の空間的な壁を取り除く。特別な条件が付かない限り、仮想計算機上で走るオブジェクトは実行時に任意のノードに移動することができる。分散システム記述言語はシステム構築に使用する核言語であり、デバイスなどの実時間プログラミング、並行動作や協調動作のための並行プログラミング、ノードにまたがっての分散プログラミングの可能な対象指向型言語である。他の既存言語で書かれたプログラムとはそのプログラムが搭載されている既存オペレイティングシステムとその上でシミュレートされている分散カーネルを介しメッセージ通信によって結合される。

（２）ロボットシステムレベルの統合性

ロボットシステムとしての統合性を高めるために、どのタイプのロボットにも必要なものとして、①ロボットサブ言語、②ロボット用データベース、③環境モデルベース、をロボットシステムの中核に据えている。

ロボットサブ言語は3節の分類に従えば、開発者用のものであり、ユーザが知る必要のない、いわばロボットの高級なドライバに相当するプログラムを作るための言語である。マニピュレータ言語、視覚言語のほかは必要になった時点で開発され追加されよう。なお、サブ言語は分散システム記述言語のマクロやスーパクラスの形で実現されるスーパセットである。ロボット用データベースをスーパクラスに持ち、実環境や仮想環境の変化によって動的に変化するのが環境モデルベースである。モデルベースの断片はネットワーク全体に分散され、様々なサブシステムはこのモデルベースによって結合されていく。

（3）ユーザを含むレベルの統合性

ユーザを含むレベルでは、ユーザとシステムが互いの壁を意識することなく、インタラクションできることが大事である。インタフェイスをサポートするためのハードウエアとして、このシステムでは5。2節で述べたようなネットワークとワークステーションを備えている。ソフトウエアに関しては、①計算機システム、②実環境とハードウエアロボット、③仮想環境とソフトウエアロボット、でのユーザとシステムの緊密なインタフェイスが考えられている。

計算機システムに関しては、並行／分散処理されているオブジェクトに対しネットワークを介した会話とデバッギングと、オブジェクトをノード間で移動させることによる負荷のチューニングが可能である。実環境とハードウエアロボットについては、マルチメディアディスプレイの立体視機能と操作教示ステックを使った遠隔操縦によって、遠隔から臨場感を持って直接作業をすることが可能である。仮想環境とソフトウエアロボットについては、グラフィクスを通して見た仮想環境において、ラフなプランの作成と修正、視覚に対する教示などが可能になると思われる。最後に、以上の機能を組み合わせ、仮想環境で動くプログラムを実環境に適用し、自動的に動かなければその部分をプログラミングしなおすことによって、段々に自律化できるのではないかと考えているが、これに関しては今後の課題である。

6。　むすび

ロボットのソフトウエアの概要を、人の近づけない環境で人に代わって作業を代行する遠隔作業ロボットの立場から紹介した。紙面の都合上、①ロボットのサーボ、②軌道の探索、③プランニング、④視覚処理、などの多くの重要な項目を割愛した。それらについては、[Computer82a],[Computer82b],[Proceedings83] などの解説記事から孫引きされたい。

遠隔作業ロボットとしては、遠隔のロボットを直接操縦するレベルから、自律的に

作業させるレベルまでいろいろなレベルが考えられる。多様な環境で、簡単な指令によって人の意図した通りに自律的に仕事を遂行するロボットは誰しもが抱く夢であり、その実現には現実に妥協することなく果敢なチャレンジが必要であろう。最後になったが、そのためには人工知能の技術やアプローチが不可欠である。

参考文献

[Computer82a] Special Issue on "Human-Computer Interaction", IEEE Computer, Vol.15, No.11 (Nov 1983)

[Computer82b] Specail Issue on "Robotics and Automation", IEEE Computer, Vol.15, No.12 (Dec 1983)

[ICAR83] Proc. of '83 International Conference on Advanced Robotics, Tokyo, (Sep 1983)

[Proceedings83] Special Issue on "Robotics", Proc. of IEEE, Vol.71, No.7, (Jul 1983)

[Bonner83] Bonner, S.: "A Comparative Study of Robot Languages", IEEE Computer, Vol.15, No.12, (Dec 1983) 82-96

[Hasegawa82] Hasegawa, T.: "A New Approach to Teaching Object Description for a Manipulator Environments", Proc. of ISIR, (May 1982) 87-98

[Lee82] Lee, C.S.G.: "Robot Arm Kinematics, Dynamics, and Control", IEEE Computer, Vol.15, No.12, (Dec 1982) 62-80

[Lozano-Perez83] Lozano-Perez, T.: "Robot Programming", Proc. of IEEE, Vol.71, No.7, (Jul 1983) 821-841

[Matsushita83] Matsushita, T. etal.: "An Attempt to Describe Visual Functions of the Hand-Eye System with a Robot Vision Language: RVL/A", Proc. of '83 ICAR, (Sep 1983) 327-334

[Mujtaba79] Mujtaba, S. etal.: "AL User's Manual", Stanford Artificial Intelligence Lab., AIM 323, (1979)

[Shimano79] Shimano, B.: "VAL: A Versatile Robot Programming and Control System", Proc. of COMPSAC79, (1979) 878-883

[Takase83] Takase, K, etal.: "Development of a Novel Manipulator of Critical Use", Proc. of '83 ICAR, (Sep 1983) 253-260

[Tayler82] Tayler, R.H. etal.: "AML: A Manufacturing Language", Robotics Research, Vol.1, No.3 (Fall 1982) 19-41

[Tayler83] Tayler, R.H. etal.: "An Integrated Robot System Architecture", Proc. of IEEE, Vol.71, No.7, (jul 1983) 842-856

[Tsukamoto83] Tsukamoto, M. etal: "Conceptual Design of a Distributed Operating System for Intelligent Robots", Proc. of '83 ICAR, (Sep 1983) 399-406

〔末広83〕末広尚士: " 作業座標系内動作記述に基づくロボット用言語" , 第22回SICE 学術講演会 (1983) 103-104

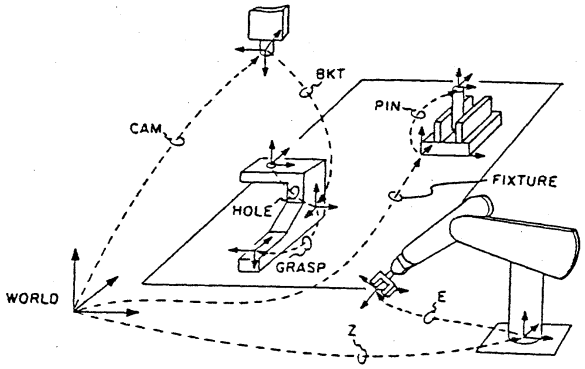
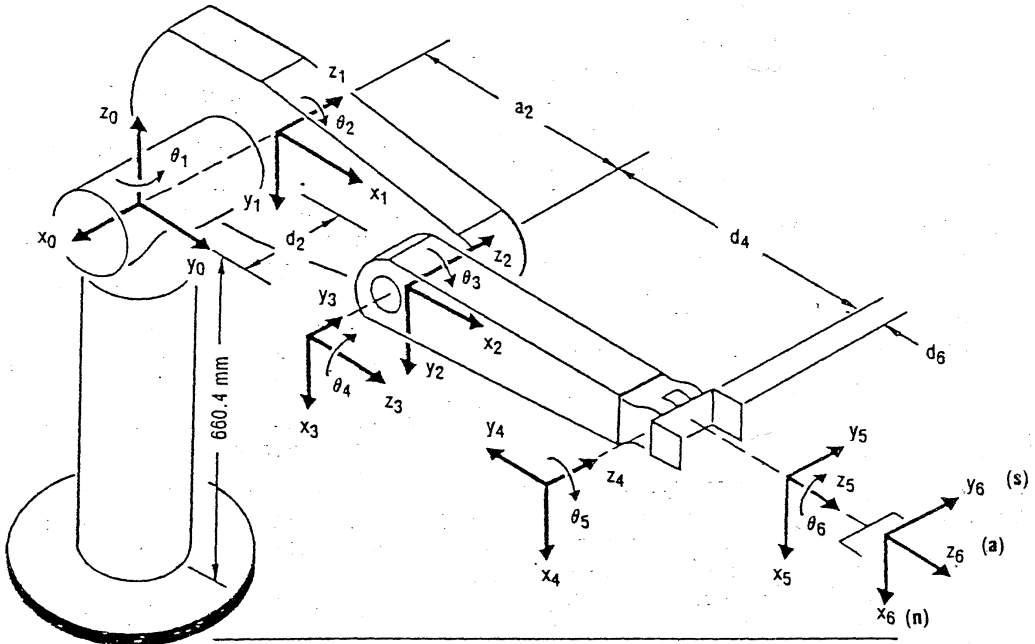


図2.1　環境表現



PUMA ROBOT ARM LINK COORDINATE PARAMETERS

JOINT i	ϑ_j	α_j	a_j	d_j	RANGE
1	90	−90	0	0	−160 to +160
2	0	0	432 mm	149.5 mm	−225 to +45
3	90	90	0	0	−45 to +225
4	0	−90	0	432.0	−110 to +170
5	0	90	0	0	−100 to +100
6	0	0	0	56.5	−266 to +266

図2.2　PUMAのリンク座標系

$$A_{i-1}^{i} = \begin{bmatrix} C\vartheta_i & -C\alpha_i S\vartheta_i & S\alpha_i S\vartheta_i & a_i C\vartheta_i \\ S\vartheta_i & C\alpha_i C\vartheta_i & -S\alpha_i C\vartheta_i & a_i S\vartheta_i \\ 0 & S\alpha_i & C\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_0^1 = \begin{bmatrix} C_1 & 0 & -S_1 & 0 \\ S_1 & 0 & C_1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_1^2 = \begin{bmatrix} C_2 & -S_2 & 0 & a_2C_2 \\ S_2 & C_2 & 0 & a_2S_2 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_2^3 = \begin{bmatrix} C_3 & 0 & S_3 & 0 \\ S_3 & 0 & -C_3 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_3^4 = \begin{bmatrix} C_4 & 0 & -S_4 & 0 \\ S_4 & 0 & C_4 & 0 \\ 0 & -1 & 0 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A_4^5 = \begin{bmatrix} C_5 & 0 & S_5 & 0 \\ S_5 & 0 & -C_5 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_5^6 = \begin{bmatrix} C_6 & -S_6 & 0 & 0 \\ S_6 & C_6 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

図2.3a　PUMAの変換マトリックス

$$\vartheta_1 = \tan^{-1}\left[\frac{\pm p_y\sqrt{p_x^2+p_y^2-d_2^2} - d_2p_x}{\pm p_x\sqrt{p_x^2+p_y^2-d_2^2} + d_2p_y}\right]$$

$$\vartheta_2 = \tan^{-1}\left[\frac{-\left(p_z(a_2+d_4S_3)+(d_4C_3)(\pm\sqrt{p_x^2+p_y^2-d_2^2})\right)}{p_z(d_4C_3)-(a_2+d_4S_3)(\pm\sqrt{p_x^2+p_y^2-d_2^2})}\right]$$

$$\vartheta_3 = \tan^{-1}\left[\frac{p_x^2+p_y^2+p_z^2-d_4^2-a_2^2-d_2^2}{\pm\sqrt{4d_4^2a_2^2-(p_x^2+p_y^2+p_z^2-d_4^2-a_2^2-d_2^2)^2}}\right]$$

$$\vartheta_4 = \tan^{-1}\left[\frac{C_1a_y-S_1a_x}{C_1C_{23}a_x+S_1C_{23}a_y-S_{23}a_z}\right]$$

$$\vartheta_5 = \tan^{-1}\left[\frac{(C_1C_{23}C_4-S_1S_4)a_x+(S_1C_{23}C_4+C_1S_4)a_y-C_4S_{23}a_z}{C_1S_{23}a_x+S_1S_{23}a_y+C_{23}a_z}\right]$$

$$\vartheta_6 = \tan^{-1}\left[\frac{(-S_1C_4-C_1C_{23}S_4)n_x+(C_1C_4-S_1C_{23}S_4)n_y+(S_4S_{23})n_z}{(-S_1C_4-C_1C_{23}S_4)s_x+(C_1C_4-S_1C_{23}S_4)s_y+(S_4S_{23})s_z}\right]$$

$$-180° \le \vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, \vartheta_5, \vartheta_6 \le 180°$$

- Degenerate case ($\vartheta_5 \approx 0$)
  ϑ_4 = current value of ϑ_4 or 0 and
  ($\vartheta_4 + \vartheta_6$) = total angle required to align the orientation.
- For a given arm configuration,
  ($\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, \vartheta_5, \vartheta_6$) is a set of solutions and
  ($\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4 + \pi, -\vartheta_5, \vartheta_6 + \pi$) is another set of solutions.

図2.3b　PUMAのアーム解

```
BEGIN
   FRAME ARRAY hole[1:3];
   FRAME base_plate;
   SCALAR i;

  {Initialization and start of the program including definition of the locations of
   the base_plate and the screw holes:
   base_plate ← FRAME(....);
   AFFIX hole[1] TO base_plate RIGIDLY AT TRANS(....);
   AFFIX hole[2] TO base_plate RIGIDLY AT TRANS(....);
   AFFIX hole[3] TO base_plate RIGIDLY AT TRANS(....);
   Screws will be defined with the z-axis pointing downward.
   Code to get the screw driver into the hand is also included. }

  {Now insert the three screws}

   FOR i ← 1 STEP 1 UNTIL 3 DO
    BEGIN
     screw ← screw_dispenser;        {Define location of new screw}
     MOVE driver_tip TO screw;       {Get a screw - not really this easy}
     AFFIX screw TO driver;
     MOVE screw_tip TO hole[i];      {Screw is just above screw hole}

     COBEGIN
       MOVE screw TO ⊙ - 0.75 * zhat * inches        {Push down with arm}
           WITH FORCE = 20 * ounces ALONG zhat OF screw
           WITH DURATION = 2.5 seconds;
       OPERATE driver                    {Drive in the screw}
           WITH VELOCITY = 200 * rpm
           WITH DURATION = 3 * seconds;
     COEND;

     UNFIX screw FROM driver             {Release the screw}
    END;
END
```

図3.1　ＡＬのプログラム例（ネジ締め）

```
 1.       SETI N.PARTS=0
 2.  100  VPICTURE
 3.       VLOCATE PART, 100
 4.       APPRO CAMERA : PART : PICK.UP, 50.
 5.       MOVES CAMERA : PART : PICK.UP
 6.       GRASP 25.
 7.       DEPART 50.
 8.       APPRO PALLET, 50.
 9.       MOVES PALLET
10.       OPENI
11.       DEPART 100.
12.       SHIFT PALLET BY 5.2, 25.4, 0
13.       SETI N.PARTS=N.PARTS+1
14.       IF N.PARTS NE 10 THEN 100
15.       STOP
```

図3.2　ＶＡＬのプログラム例（部品のパレタイジング）

```
GRASP: SUBR(GRIPPER_OPENING,<MIN_OFS,MAX_OFS>,F);
   TOGO:   NEW REAL;
   FMONS: NEW APPLY($MONITOR,PINCH_FORCE(F));
    CLEANUP($CLN);

    MOVE(GRIPPER,GRIPPER_OPENING+MIN_OFS,FMONS);

    IF QMONITOR(FMONS(1)) EQ O THEN
       BEGIN                                  -- Left finger did not hit.
       IF QMONITOR(FMONS(2)) EQ O THEN RETURN('TOO SMALL');
       TOGO = GRIPPER_OPENING+MIN_OFS-QPOSITION(GRIPPER);
       DMOVE(XYZ#<GRIPPER>,
              (TOGO/2*(HANDFRAME)(2,2))#<TOGO>,FMONS(1));
       END
    ELSE IF QMONITOR(FMONS(2)) EQ O THEN
       BEGIN                                  -- Right finger did not hit.
       TOGO = GRIPPER_OPENING+MIN_OFS-QPOSITION(GRIPPER);
       DMOVE(XYZ#<GRIPPER>,
              (-TOGO/2*HANDFRAME)(2,2))#<TOGO>,FMONS(2));
       END;

    RETURN( IF QPOSITION(GRIPPER)
                  LE GRIPPER_OPENING+MAX_OFFSET THEN'OK'
          ELSE 'TOO BIG');

  CLN: SUBR;                              -- Called whenever GRASP exits
       ENDMONTIOR(FMONS);                 --  to terminate force monitors
       END;
    END;
```

図3.3　AMLのプログラム例（つかみ）

```
RECOGNIZEPART: SUBR (FEATURESPACE);
    -- Acquires an image, derives feature vector, and returns
    -- index of nearest cluster in FEATURESPACE.
    FEATUREVEC: NEW PARTFEATURES;  -- Derive feature vector
    NCLUSTERS:   NEW AGGSIZE(FEATURESPACE);  -- Number of clusters
    DISTANCES:   NEW FEATUREDISTANCES (FEATUREVEC, FEATURESPACE);

    RETURN ( MININDICES (DISTANCES) );

    MININDICES: SUBR (A);
      -- Returns indices of minimum elements of aggregate A
      N: NEW AGGSIZE(A);  -- Number of elements in A
      RETURN (SELECT(-N EQ APPLY($+,(N OF A) LE A)),IOTA(N));
    END;  -- MININDICES

END;  -- RECOGNIZEPART
```

図3.4　AML／Vのプログラム例（部品識別）

```
1   (DE EXP-PROG NIL
2     (MOVE IN 'COVER-GRIP
3       (DO 'HAND1
4         (OPEN 120)
5         (STR '(0 0 -30 0 0 0))
6         (WITH ((COMPL '(Z) '(0.02))) (STR '(0 0 0 0 0 0)))
7         (OPEN 80)
8         (AFFIX 'COVER TO 'HAND1)
9         (STR '(0 0 -150 0 0 0))))
10    (MOVE IN 'BOX-CORNER
11      (DO 'COVER-CORNER
12        (STR '(5 5 10 0 0 0))
13        (WITH ((COMPL '(Z) '(0.02))) (STR '(0 0 0 0 0 0)))
14        (WITH ((COMPL '(Z) '(0.02))) (FORCE '(X Y) '(0 0)))
15          (STR '(0 0 -10 0 0 0 ))))
16      (DO 'HAND1
17        (OPEN 120)
18        (UNFIX 'COVER FROM 'HAND1)
19        (STR (HERE+ '(0 0 50 0 0 0)))))
20    (FOR (I 1 6)
21      (MOVE IN (_ 'BOLT I)
22        (DO 'HAND1
23          (STR '(0 0 -20 0 0 0))
24          (STR '(0 0 0 0 0 0))
25          (OPEN 0)
26          (AFFIX (_ 'BOLT I) TO HAND1)
27          (STR '(0 0 -100 0 0 0))))
28      (MOVE IN (_ 'AHOLE I)
29        (DO (_ 'BOLT-TIP I)
30          (STR '(0 0 -30 0 0 0))
31          (WITH ((COMPL '(Z) '(0.02))) (STR '(0 0 0 0 0 0))))
32        (DO 'HAND1
33          (FOR (J 1 7)
34            (WITH ((FORCE '(X Y Z) '(0 0 0.01)))
35              (STR (HERE+ '(0 0 0 0 0 150))))
36            (STR (HERE+ '(0 0 0 0 0 0)))
37            (OPEN 0)
38            (UNFIX (_ 'BOLT I) FROM 'HAND1)
39            (STR (HERE+ '(0 0 0 0 0 -150)))
40            (OPEN 0)
41            (AFFIX (_ 'BOLT I) TO 'HAND1))
42          (OPEN 30)
43          (UNFIX (_ 'BOLT I) FROM 'HAND1)
44          (AFFIX (_ 'BOLT I) TO (_ 'HOLE I))
45          (STR '(0 0 -100 0 0 0))))))
```

```
1   プログラム開始．ＰＥＴＬの関数定義による．EXP-PROGはプログラム名．
2   COVER-GRIPを作業座標系として・・・
3   HAND1(手先) を・・・
4   120 ㎜  に開く．
5   COVER-GRIPの 30 ㎜  上に動かす．
6   縦方向のコンプライアンスを 0.02 kgw/㎜としてCOVER-GRIPと一致させる．
7   80 ㎜ に開く（閉じる）．
8   データベース上でCOVER をHAND1 に固定する．
9   COVER-GRIPの 150 ㎜ 上に動かす．
10  BOX-CORNERを作業座標系として・・・
11  COVER-CORNERを・・・
12  BOX-CORNERの斜め上 "(5 5 10 0 0 0)" に持っていく．
13  縦方向のコンプライアンスを 0.02 kgw/㎜  としてBOX-CORNERと一致させる．
14  縦方向のコンプライアンスを 0.02 kgw/㎜  とし、横方向の力を零とし、
15   10 ㎜  押しつける．
16  HAND1 を・・・
17  120 ㎜  に開く．
18  COVER をHAND1 から切離す．
19  現在位置"(HERE+"より 150 ㎜ 上に持ちあげる．
20  以下をＩが１から６までくり返す．
21  BOLT_I "(_ 'BOLT I)"を作業座標として・・・
22  HAND1 を・・・
23  BOLTの上 20 ㎜  に動かす．
24  BOLTの所に動かす．
25  0 ㎜  に開く（閉じる）．
26  BOLTをHAND1 に固定する．
27  BOLTの上 100 ㎜ に動かす．
28  AHOLE I を作業座標として・・・
29  BOLT-TIP_Iを・・・
30  AHOLE の上 30 ㎜  に持っていく．
31  縦方向のコンプライアンスを 0.02 kgw/㎜  としてAHOLE へ持っていく．
32  HAND1 を・・・
33  以下Ｊが１から７までくり返す．
34  横方向の力は０ kgw  縦方向の力は 0.01 kgw として、
35  現在位置からＺ軸まわりに 150度まわす．
36  力を抜いて現在位置でとまる．
37  30 ㎜ に開く．
38  BOLTをHAND1 から切離す．
39  現在位置からＺ軸まわりに-150度まわす．
40  閉じる．
41  BOLTをHAND1 に固定する．（ここまでＪのくり返し．）
42  30 ㎜ に開く．
43  BOLTをHAND1 から切離す．
44  BOLTをHOLEに固定する．
45  上 100 ㎜ に動かす．（ ここまでＩのくり返し。）
```

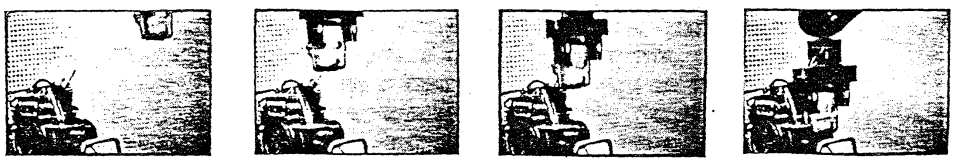
図3.5　プログラム例（箱のふたしめ）

```
( CREATE `ENGINE )                ; INITIALIZE DESCRIPTION OF " ENGINE ".
 ( EVIEW 1 )
  ( SETTRAN `TRAN1 0 0 0 )        ; DEFINITION OF THE FRAME OF 1ST PART.
  ( EPART `LINE2 `PORT TRAN1 )    ; START EDITING OF THE EXHAUST PORT APPEARANCE
   ( SETPAS `DTH 500 )            ; APPARANCE OF PORT IS LINE2.
   ( SETPAS `LNGT 10 )            ; SETPAS SETS PARAMETER VALUE.
   ( SETPAR `AV 600 600 500 )     ; SETPAR SETS A  RANGE OF PARAMETER VALUE.
   ( SETPAS `DL 500 )             ; ALL THESE PARAMETERS ARE USED BY BASIC
   ( SETPAR `GAP 2000 2000 1000 ) ; FUNCTIONS FOR PROFILE ANALYSIS.
   ( SETPAR `E1 1200 600 600 )    ;
   ( SETPAR `E2 1200 600 600 )    ;
  ( SETTRAN `TRAN2 -1500 2000 0 ) ; SET FRAME PARAMETER OF 2ND PART.
  ( EPART `STEP2 `COVER TRAN2 )   ; START DESCRIPTION OF THE ENGINE COVER.
              --                  ; " ENGINE " IS COMBINATION OF " PORT " AND
              --                  ; " COVER ".
```

図3.6　ＲＶＬのエンジンのモデル記述

```
( TAKRD )                   ; INPUT PROFILE DATA.
( PPRD )                    ; PREPROCESSING ( SEGMENTATION ).
( FIND `ENGINE )            ; VERIFY GASOLINE ENGINE EXISTENCE.
( MEASURE `ENGINE `PCENT )  ; MEASURE CENTER POSITION OF THE PROFILE.
     ...                    ; 1ST ELEMENT(PORT) IS SELECTED.
     ...                    ; DESTINATION OF HAND IS SET.
( MEASURE `ENGINE `PFRAME ) ; MEASURE FRAME AXIS DIRECTION (OF PORT).
     ...                    ; ROTATION OF WRIST IS SET.
( MOVEARM X Y Z A B C )     ; MOVE AND FIT MUFFLER TO PORT.
     ...
```

(a)



(b)

図3.7　ＲＶＬによるエンジンマフラの装着

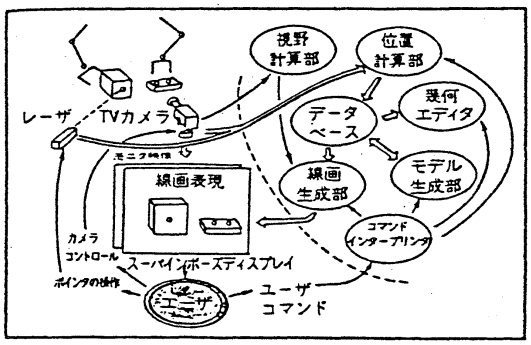


図4.1　環境教示システムの構成

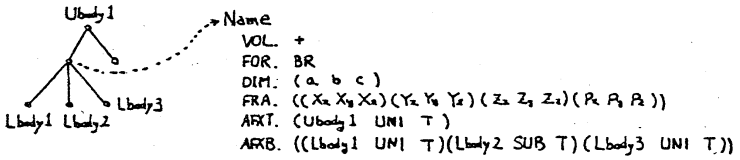


図4.2　対象物のデータセル表現

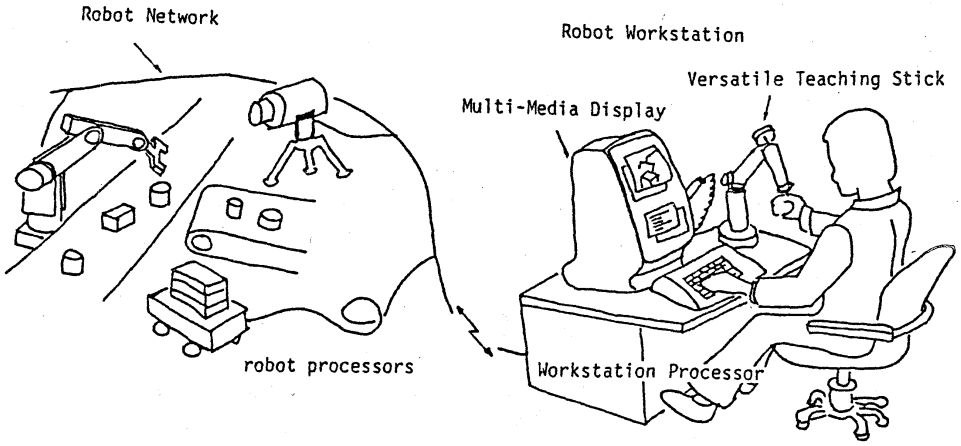


図5.1　ロボット統合システムの概念図

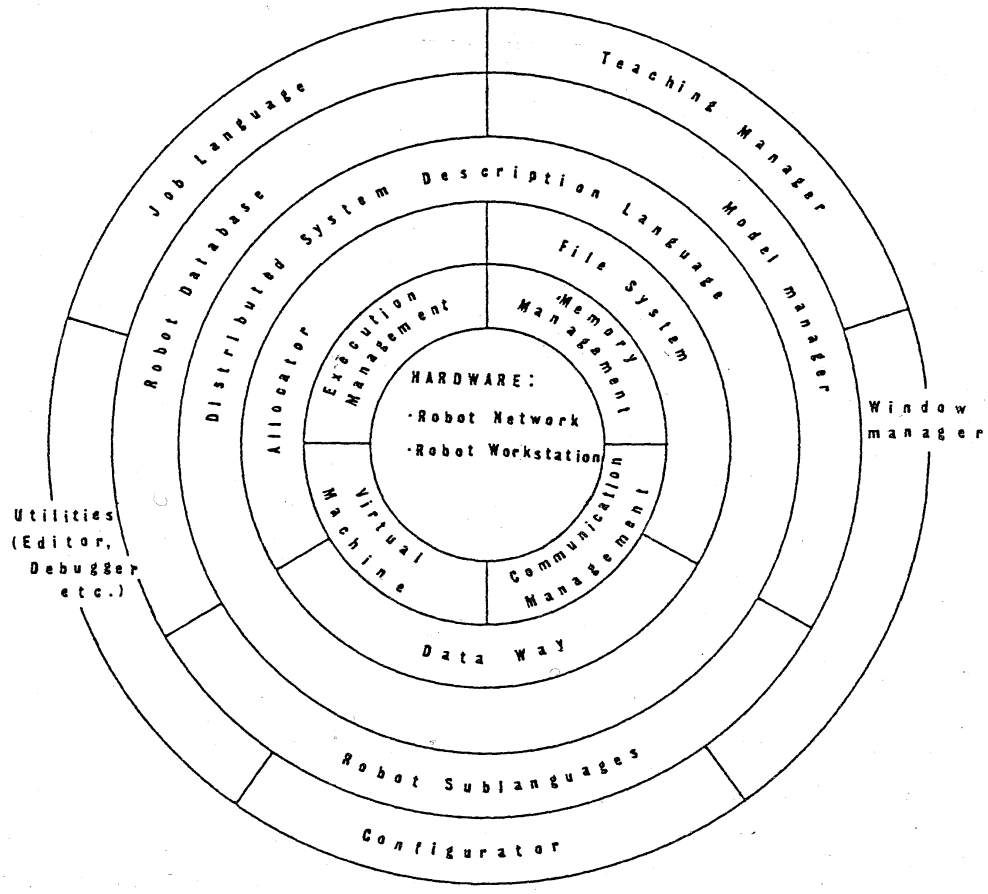


図5.2　ロボットオペレイティングシステムの機能階層